Ciratefi: An RST-Invariant Template Matching with Extension to Color Images

Sidnei Alves de Araújo[a,b] and Hae Yong Kim[a,*]
[a] *Escola Politécnica – Universidade de São Paulo, Av. Prof. Luciano Gualberto, tr. 3, 158, São Paulo, SP, Brazil*
[b] *Diretoria dos Cursos de Informática – Universidade Nove de Julho, Rua Diamantina, 302, São Paulo, SP, Brazil*

Abstract. Template matching is a technique widely used for finding patterns in digital images. A good template matching should be able to detect template instances that have undergone geometric transformations. In this paper, we proposed a grayscale template matching algorithm named Ciratefi, invariant to rotation, scale, translation, brightness and contrast and its extension to color images. We introduce CSSIM (color structural similarity) for comparing the similarity of two color image patches and use it in our algorithm. We also describe a scheme to determine automatically the appropriate parameters of our algorithm and use pyramidal structure to improve the scale invariance. We conducted several experiments to compare grayscale and color Ciratefis with SIFT, C-color-SIFT and EasyMatch algorithms in many different situations. The results attest that grayscale and color Ciratefis are more accurate than the compared algorithms and that color-Ciratefi outperforms grayscale Ciratefi most of the time. However, Ciratefi is slower than the other algorithms.

Keywords: Template matching, RST-invariance, color invariance.

1. Introduction

Template matching is a classical problem in computer vision. It consists in detecting a given query template Q in a digital image A. This task becomes more complex with the invariance to rotation (R), scale (S) and translation (T), and robustness to brightness (B) and contrast (C) changes. We define that two images $\boldsymbol{x}$ and $\boldsymbol{y}$ are equivalent under brightness/contrast variation if there are a brightness correction factor α and a contrast correction factor $\beta>0$ such that $\boldsymbol{y} = \beta\boldsymbol{x} + \alpha\boldsymbol{1}$, where $\boldsymbol{1}$ is the matrix of 1's.

Some approaches achieve RST-invariance using detection of interest points and edges, including: generalized Hough transform [4]; geometric hashing [20]; graph matching [21]; and curvature scale space [28].

Ullah and Kaneko [37] present a RTBC-invariant grayscale template matching based on orientation gradient histograms. This algorithm was improved by Marimon e Ebrahimi [25] by using integral histograms. Choi and Kim [9] present another interesting algorithm for RTBC-invariant template matching. Cyganek [11] present a simple and fast technique for detecting circular road signs. Raftopoulos et al. [30] present a biologically inspired shape classifier that may shed light in the mechanism of biological template matching. Sajjanhar et al. present an interesting rotation-invariant shape descriptor method using spherical harmonics [32].

The recently developed algorithms based on scale and rotation-invariant keypoints and local features, like SIFT (Scale Invariant Feature Transform) [24], GLOH (Gradient Location and Orientation Histogram) [26] and SURF (Speeded Up Robust Features) [6] have been widely used for image matching tasks. SIFT is very popular and has been proven to be the one of the most efficient methods to extract invariant features from images. It extracts some scale-invariant keypoints and computes their associated features based on local gradient orientations. Then, it finds the correspondences between the keypoints of Q and A based on the distances of the features. SIFT (followed by a Hough transform to identify clusters be-

* Corresponding author. E-mail: hae@lps.usp.br

longing to a single object) can be used for object recognition or template matching.

Fourier-Mellin transform is a popular technique for RST-invariant image registration, where image Q may appear rotated, scaled and translated in A [31, 8]. However, this technique supposes implicitly that the non-intersecting areas of Q and A are small. So, it cannot be directly applied for template matching, where template Q and image A usually have large non-intersecting areas. If Fourier-Mellin is applied in this case, the correlation peaks become weak, making it difficult to detect the geometric transformation parameters.

There are also some commercial RSTBC-invariant template matchings, such as Open eVision EasyMatch[1]. However, as they are commercial software we do not know exactly which algorithm is implemented inside.

In this paper, we present a template-matching algorithm, named Ciratefi (Circular, Radial and Template-Matching Filter), invariant to rotation, scale, translation, brightness and contrast (RSTBC). The goal of Ciratefi is to find all occurrences of a query image Q in an image to be analyzed A, with respective orientation angle and scale. In some applications, the number of instances of Q in A may be known, and this information is very useful to the algorithm. Ciratefi algorithm consists of three cascaded filters that successively exclude pixels that have no chance of matching the template from further processing. Ciratefi does not require previous "simplification" of A and Q that discards grayscale information, like detection of edges, detection of interest points or segmentation/binarization. These image simplifying operations seems to be noise-sensitive and prone to errors. Experimental results show that the absence of "simplification" makes Ciratefi very robust to some common image distortions such as blurring and JPEG-compression.

We also propose an extension of Ciratefi named color-Ciratefi that takes into account the color information. Most existing template matching techniques were designed for gray-level images, not taking into account the power of color. The main problem of color template matching is the color constancy, that is, how to extract color information that remains constant with the illumination change. Color is not an intrinsic property of objects. Instead, the apparent color of objects depends on the spectral composition of the illuminant, the reflecting properties of their surfaces and the color of the environment. Changes in illumination can cause changes in object colors acquired by a camera, worsening the performance of pattern recognition algorithms that use color information [27].

Since color constancy is an open problem, it has been an object of intense study and in the last two decades many color-based methods, invariant color models and perceptual distance measures have been proposed [10, 5]. However, some studies have questioned the effectiveness and usefulness of the proposed color invariants. For example, Funt et al. [13] investigated several methods used in color-based object recognition and concluded that all tested methods are insufficient to deal with color constancy problem. Schaefer [34] investigated the usefulness of color invariants in image retrieval. He concluded that color invariants are not always useful for image retrieval. He also suggests that some prior knowledge of the application domain is necessary to improve the performance of invariants.

Several proposed color-based matching algorithms use only global color histograms of images ignoring the spatial information [35, 12, 16]. These algorithms are more suitable for image retrieval applications or object recognition in a simple background.

Tsai and Tsai [36] present a technique for matching colored objects that is somehow similar to ours. The main drawback of this technique is the lack of invariance to scale changes.

Geusebroek et al. [14] developed an important set of color invariant features based on Gaussian derivative to deal with illumination changes, shadow, highlights and noise. These set of invariants was embedded in the SIFT, by Burghouts and Geusebroek [7], yielding powerful color invariant descriptors. Among them are W-color-SIFT, H-color-SIFT and C-color-SIFT. Actually, many color invariants reported in the literature have been plugged in the SIFT, generating many other color-based SIFT descriptors such as CSIFT [1], HSV-SIFT, Hue-SIFT, OpponentSIFT, W-SIFT, rgSIFT, Transformed color SIFT [33] and SIFT-CCH [2].

Color-Ciratefi has the same structure of Ciratefi, but was designed to deal with color images. To achieve controlled robustness to illumination changes, we proposed a new similarity measure, named CSSIM, which is a weighted geometric mean between SSIM (structural similarity) [38] and the Euclidean distance of components a* and b* from

[1] http://www.euresys.com/Products/MachineVisionSoftware/MachineVision.asp

CIE L*a*b* color space, used as similarity of chromaticity.

In order to attest the accuracy of Ciratefi and color-Ciratefi, we conducted several experiments comparing their results with the results obtained by SIFT, C-color-SIFT and EasyMatch algorithms. We tested images with few textures, large illumination variation, blur, JPEG compression and viewpoint variations. The results show that Ciratefi and color-Ciratefi are more accurate than the compared algorithms and also that Color-Ciratefi outperforms grayscale Ciratefi in most situations.

2. Original Ciratefi[2]

The objective of grayscale Ciratefi algorithm is to find a query template image Q in a larger image to analyze A, invariant to rotation, scaling, and translation and with controlled robustness to brightness and contrast changes. Ciratefi consists of three cascaded filters: Cifi, Rafi and Tefi. Each filter successively excludes pixels that have no chance of matching the template. Moreover, Cifi and Rafi also compute respectively the scale and rotation angle.

Actually, Ciratefi uses a set of discrete angles and scales. To avoid that a small misalignment may cause a large mismatching, a low-pass filter (for example, the Gaussian filter) may smooth both images A and Q. This low-pass filtering lessens the errors introduced by using discrete scales and angles.

2.1. Correlation coefficient

The original Ciratefi uses the correlation coefficient (also known as normalized correlation) in each Ciratefi step to evaluate how well Q matches (in brightness/contrast-invariant sense) a region of A around a pixel (x, y). Let $\mathbf{v}$ be the vector that represents the mean grayscales of certain parts of Q and $\mathbf{w}$ the corresponding vector of the parts of $A(x, y)$. Then, the correlation coefficient is defined:

$$X(\mathbf{v},\mathbf{w}) = \frac{\tilde{\mathbf{v}}\tilde{\mathbf{w}}}{\|\tilde{\mathbf{v}}\|\|\tilde{\mathbf{w}}\|}. \qquad (1)$$

where $\tilde{\mathbf{v}} = \mathbf{v} - \bar{\mathbf{v}}$ is the mean-corrected vector and $\bar{\mathbf{v}}$ is the mean of $\mathbf{v}$ (similar definitions are applicable to $\mathbf{w}$). Division by zero must be avoided certifying that $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{w}}$ are not null vectors. The correlation coefficient between two non-zero vectors always ranges from -1 to +1 and is BC-invariant. If the correlation is greater than some threshold $t \geq 0$, we consider that Q matches $A(x, y)$. This comparison may be signed $X(\mathbf{v},\mathbf{w}) \geq t$ (in this case, a black-white reversed negative instance will not match) or absolute $|X(\mathbf{v},\mathbf{w})| \geq t$ (either negative or positive instance will match).

2.2. First Filter: Cifi

The first filter, called Cifi (Circular sampling Filter) uses the projections of the images A and Q on a set of circular rings (Figure 1) to detect the "first grade candidate pixels" and their "best matching scales". Given an image B, let us define the circular sampling as the average grayscale of the pixels of B situated at distance r from pixel (x, y):

$$S_B^{\Omega}(x,y,r) = \frac{1}{2\pi r}\int_0^{2\pi} B(x + r\cos\theta, y + r\sin\theta)\, d\theta \quad .(2)$$

We use the superscript Ω to indicate circular sampling.

Given the template Q and the set of n scales $s_0, s_1, \ldots, s_{n-1}$, the image Q is resized to each scale s_i, obtaining the resized templates $Q_0, Q_1, \ldots, Q_{n-1}$. Then, each resized template Q_i is circularly sampled at a set of circle rings with l predefined radii $r_0, r_1, \ldots, r_{l-1}$, yielding a 2-D table of multi-scale rotation-invariant features with n rows (scales) and l columns (radii):

$$C_Q[i,k] = S_{Q_i}^{\Omega}(x_0, y_0, r_k), \ 0 \leq i < n \text{ and } 0 \leq k < l \quad (3)$$

where (x_0, y_0) is the central pixel of Q. In small scales, some of the outer circles may not fit inside the resized templates. These circles are represented by a special value in table C_Q (say, -1) and are not used to compute the correlations.

Given the image to analyze A, we build the 3-D image that contains l circular projections for each pixel (x, y):

$$\begin{aligned} C_A[x,y,k] &= S_A^{\Omega}(x,y,r_k), \\ 0 \leq k < l, &(x,y) \in \text{domain}(A) \end{aligned} \quad . \qquad (4)$$

Cifi uses matrices C_Q and C_A to detect the circular sampling correlation at the best matching scale for each pixel (x, y):

$$X_{A,Q}^{\Omega}(x,y) = \underset{i=0}{\overset{n-1}{\text{MAX}}}\left[X(C_Q[i], C_A[x,y])\right], \qquad (5)$$

where $X(C_Q[i], C_A[x,y])$ is the correlation coefficient between vectors $C_Q[i]$ and $C_A[x,y]$. A pixel (x, y) is classified as a "first grade candidate pixel" if $X_{A,Q}^{\Omega}(x,y) \geq t_1$ for some threshold t_1. The appropri-

[2] A preliminary work was published in [18].

ate value for t_1 depends on the application. Assigning small value for t_1 makes the algorithm slower, and assigning large value for t_1 decreases the algorithm's accuracy. In section 4, we explain this parameter can be automatically chosen. The "best matching scale" of a first grade candidate pixel (x, y) is the argument that maximizes the correlation:

$$G^{\Omega}_{A,Q}(x,y) = \underset{i=0}{\overset{n-1}{\text{ARGMAX}}}\left[X(C_Q[i], C_A[x,y])\right]. \quad (6)$$

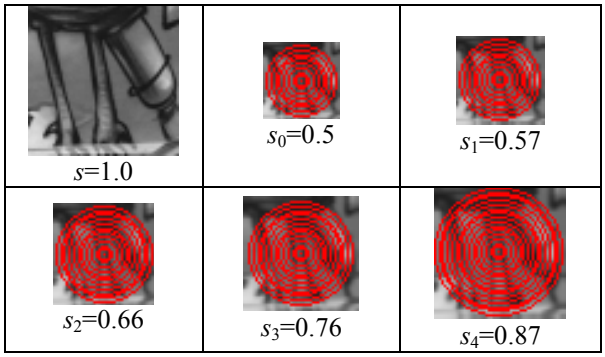

Fig. 1. The original query image and the circular projections at different scales.

2.3. Second Filter: Rafi

The second filter, called Rafi (Radial sampling Filter), uses the projections of images A and Q on a set of radial lines (Figure 2) to upgrade some of the first grade candidate pixels to the second grade. The pixels that are not upgraded are discarded. It also assigns the "best matching rotation angle" to each second grade candidate pixel. Given an image B, let us define the radial sampling as the average grayscale of the pixels of B located on the radial line with one vertex at (x, y), length λ and inclination α:

$$S^{\Phi}_B(x,y,\lambda,\alpha) = \frac{1}{\lambda}\int_0^{\lambda} B(x + t\cos\alpha, y + t\sin\alpha)\, dt \quad (7)$$

We use the superscript Φ to indicate radial sampling.

Given the template Q and the set of m angles α_0, α_1, ..., α_{m-1}, Q is radially sampled using r_{l-1}, the radius of the largest sampling circle that fits inside Q, yielding a vector with m features:

$$R_Q[j] = S^{\Phi}_Q(x_0, y_0, r_{l-1}, \alpha_j),\ 0 \le j < m \quad (8)$$

where (x_0, y_0) is the central pixel of Q.

For each first grade candidate pixel (x, y), A is radially sampled at its probable scale $i = G^{\Omega}_{A,Q}(x,y)$:

$$R_A[x,y,j] = S^{\Phi}_A(x,y,s_i r_{l-1}, \alpha_j),\ 0 \le j < m\ . \quad (9)$$

where $s_i r_{l-1}$ is the radius of the scaled template Q_i.

At each first grade candidate pixel (x, y), Rafi uses vectors $R_A[x,y]$ and R_Q to detect the radial sampling correlation at the best matching angle:

$$X^{\Phi}_{A,Q}(x,y) = \underset{j=0}{\overset{m-1}{\text{MAX}}}\left[X\left(R_A[x,y], \text{cshift}_j(R_Q)\right)\right], \quad (10)$$

where "cshift$_j$" means circular shifting (or element-wise rotation) j positions of the argument vector. A first grade pixel (x, y) is upgraded to the second grade if $X^{\Phi}_{A,Q}(x,y) \ge t_2$ for some threshold t_2. In section 4, we explain how to select this parameter automatically. The probable rotation angle at a second grade candidate pixel (x, y) is the angle that maximizes the correlation:

$$G^{\Phi}_{A,Q}(x,y) = \underset{j=0}{\overset{m-1}{\text{ARGMAX}}}\left[X\left(R_A[x,y], \text{cshift}_j(R_Q)\right)\right]. \quad (11)$$

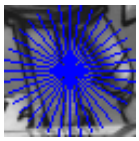

Fig. 2. Radial projections at the selected scale.

2.4. Third Filter: Tefi

The third filter, called Tefi (Template matching Filter), computes the correlation coefficient between the neighborhood of each second grade candidate pixel and the template scaled and rotated using the scale and angle determined respectively by Cifi and Rafi.

Tefi first resizes and rotates template Q to all m angles and n scales and stores them in a table named T_Q. Let (x, y) be a second grade candidate pixel, with its probable scale $i = G^{\Omega}_{A,Q}(x,y)$ and probable angle $j = G^{\Phi}_{A,Q}(x,y)$. Tefi computes the correlation coefficient between the image A at pixel (x, y) and $T_Q[i, j]$ (the template image Q at scale s_i and angle α_j). Actually, to make the algorithm more robust, Tefi tests scales i-1, i, i+1 and angles j-1, j, j+1 (in this case, subtraction and addition are computed modulus m) and takes the greatest correlation coefficient. If the greatest coefficient is above some threshold t_3, the template is considered to be found at pixel (x, y), at the scale and the angle that yielded the greatest corre-

lation. If the user knows that Q appears only once in A, the threshold t_3 is not used. Instead, a single pixel with the highest correlation is chosen.

2.5. Considerations

Cifi, Rafi and Tefi cannot be executed in different order, because Cifi detects the probable scale, to be used by Rafi, and Rafi detects the probable angle, to be used by Tefi. Rafi cannot be executed without the probable scale, and Tefi cannot be executed without the probable scale and angle.

Some query images may have "ambiguous" scale and/or rotation angle. Consider the query image depicted in Figure 3. All the circular and radial projections yield the same average grayscale. So, Ciratefi cannot search for this image. However, even in this case, Ciratefi can search for an off-centered subimage of this query image.

Fig. 3. A query image with ambiguous scale and rotation angle.

Let us make the following considerations to show that, under some assumptions, if a rotated, scaled or brightness/contrast-changed (but not deformed) instance of non-ambiguous Q appears in A, it will always be detected by Ciratefi. Let us suppose that Cifi uses enough number of circular projections and the images were low-pass filtered so that any scaled instances of Q in A yield correlations almost equal to one. In this case, it is clear that Cifi will never make a false negative error. However, Cifi can still make false positive errors, because different templates can be mapped into the same circular projections. Similarly, supposing that Cifi has computed the correct probable scale and using enough number of radial projections, Rafi will never make a false negative error, though it can make false positive errors. Finally, supposing Q is non-ambiguous and that the scale and angle were correctly computed, Tefi will never make any (false negative or positive) error, because it compares all the pixels of Q and of the corresponding instance in A to compute the correlation coefficient. An instance of Q in A may pass undetected by Ciratefi only if Q is ambiguous or if the chosen thresholds are too high to discard a true matching.

The template matching proposed by Lin and Chen [23] uses circular (or ring) projection together with parametric template to obtain RST-invariance. It has no secondary filter as Rafi or Tefi. As we discussed above, it is not possible to avoid false positive errors using only circular projections (or any template matching that uses a set of a few local features), because different templates can be mapped into the same features.

3. Accuracy of the Original Ciratefi

We compare original Ciratefi[3] with SIFT (Scale Invariant Feature Transform) proposed and implemented by Lowe[4] and EasyMatch 1.0, a template matching tool of Euresys Open eVision.

Lowe's SIFT implementation finds the keypoint correspondences, but does not find the template matching locations. To find Q in A using the keypoint correspondences, we implemented the generalized Hough transform [4] to identify clusters, as suggested by Lowe. Although both SIFT and Ciratefi can be used for image matching, they are quite different. We enumerate some differences below:

1. SIFT is based on keypoints and local gradient orientations while Ciratefi compares directly grayscales of regions of Q and A. This makes Ciratefi more reliable than SIFT when the images are blurred, have large areas with constant grayscales, suffer JPEG compression, have few textures or suffer large brightness/contrast changes. SIFT is more reliable than Ciratefi when the images have many small textures. A small misalignment of the tiny textures may decrease the correlation used by Ciratefi to find the template, decreasing its accuracy.
2. Template matching using SIFT followed by Hough transform is robust to partial occlusions, while Ciratefi alone is not. However, Ciratefi (as well as any template matching algorithm) can become robust to partial occlusions by taking some sub-templates of Q, finding them all in A, and combining the results by a Hough transform. This idea was used in [19].

[3] Executable Ciratefi implementation is available at http://www.lps.usp.br/~hae/software/cirateg

[4] http://www.cs.ubc.ca/~lowe/keypoints/

3.1. Preliminary experiments

We made some preliminary experiments and the results are depicted in Table 1. SIFT and EasyMatch use square query images and Ciratefi uses only the circular regions inscribed in the square query images. SIFT and Ciratefi searched for the templates without knowing how many instances occur in image *A*. We specified in the Hough transform (that follows SIFT) that a matching must have at least a cluster of three keypoint correspondences, as suggested by Lowe [24]. On the other hand, we informed EasyMatch the exact number of occurrences of the template. This gives a comparative advantage to EasyMatch and makes the numbers of false positive and negative errors to be always the same. Table 2 presents a simplification of Table 1, where we present the accuracy computed as Hits/(Hits+FP+FN). Note that this formula is slightly different from the conventional accuracy (TP+TN)/ (TP+TN+FP+FN), because there is no TN in our examples.

a) Toy shapes

We took some simple toy shapes, scattered them on the floor, and took 70 photos with different zooms. Then, we extracted 5 query images with 51×51 pixels and searched for them in the 70 images. In each image to analyze, there are two instances (with different grayscales but the same shape) of each query image, resulting in 700 total searchings. Ciratefi found correctly all the 700 instances, without any false negative or false positive (accuracy 100%). Figure 4(a) depicts one of the outputs of Ciratefi, obtained by concatenating the results of the 5 searchings. As the query images have few textures, we can anticipate that SIFT will perform poorer than Ciratefi. Indeed, SIFT hit only 564 searchings, missed 136 and made 4 false positive errors (accuracy 80%). Due to the impossibility to run automatized experiments, we used EasyMatch to search only for the "frog" shapes. It hit 131 searchings (out of 140 possible matchings) and missed 9 (accuracy 88%).

b) McDonald's logotype

We searched for the McDonald's logotype in 60 natural images where the logo appears 116 times (Figure 4(b)). Ciratefi has more hits (114) than SIFT (54) or EasyMatch (101), with accuracies respectively 97%, 39% and 77%. This query image also has few textures and so SIFT's performance is poor.

c) H-shaped Buildings

We searched for H-shaped popular dwelling buildings in the 15 satellite images provided by Google Earth (Figure 4(c)). The query building appears 187 times in the 15 images. Again, Ciratefi has more hits (171) than SIFT (61) or EasyMatch (118), with accuracies respectively with 83%, 21% and 46%.

d) Memory cards

We searched for 12 memory cards in 3 sets of 10 images to analyze. This problem is appropriate for SIFT, because the query images have many textures. In set A, the cards are only rotated. In set B, the cards are rotated and scaled. In set C, the cards are rotated with partial occlusions (Figure 4(d)).

Tables 1 and 2 present the results. In set A, Ciratefi and SIFT made zero errors, closely followed by EasyMatch (accuracies 100%, 100% and 97%). In set B, SIFT has the best accuracy (98%), closely followed by Ciratefi (96%). EasyMatch has substantially lower accuracy (38%). In set C, SIFT fared much better (accuracy 100%) than Ciratefi (73%) and EasyMatch (63%). Note that we are comparing SIFT followed by Hough transform against Ciratefi and EasyMatch alone. The former is robust against partial occlusions while the latters are not. This explains SIFT's clear superiority in set C.

3.2. Mikolajczyk's image database

We made more comparison between Ciratefi and SIFT using the 8 sets of natural images provided by Mikolajczyk[5] (Figure 5). This database is adequate for testing the image searching algorithm's robustness to focus blur, viewpoint changes (perspective), camera aperture (brightness/contrast change), JPEG compression, zoom and rotation. Each set is a sequence of 6 progressively distorted images, totalizing 48 images. For each set, we extracted 50 square query images with 41×41 pixels uniformly distributed within the first image and searched for them in the 6 images. Thus, each experiment consisted of 300 searchings. SIFT and EasyMatch use all the square query image, while Ciratefi uses only the circular region inscribed in the square query image.

[5] http://www.robots.ox.ac.uk/~vgg/research/affine/

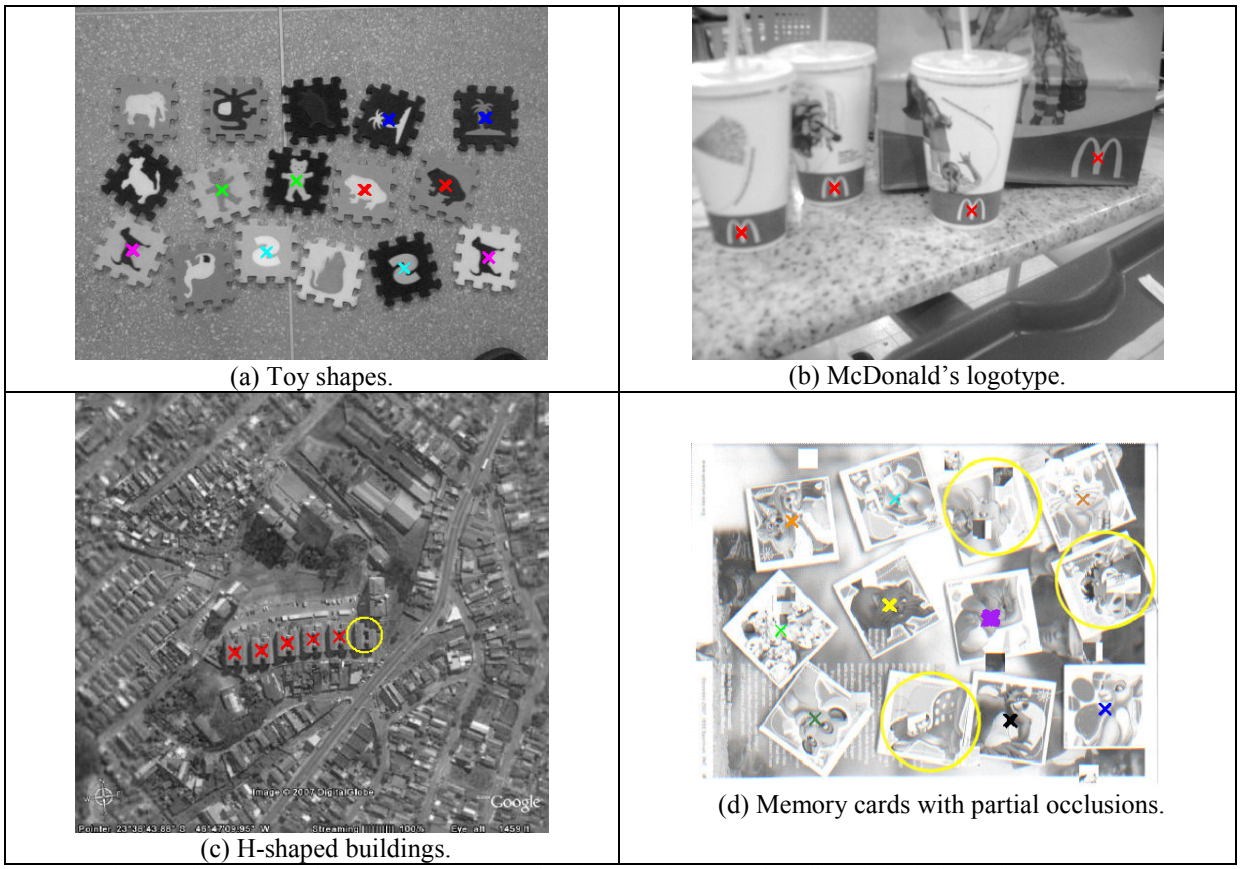

Fig. 4. Preliminary experiments to compare Ciratefi with SIFT and EasyMatch. Yellow circles indicate false negative errors.

Table 1
Hits and error rates of Ciratefi, SIFT and EasyMatch in preliminary experiments. * means that the experiment was not done.

		toy shapes								memory game		
		frog	bear	tree	letter-S	dog	total	McDonald	buildings	A	B	C
Possible matchings		140	140	140	140	140	700	116	187	120	120	120
Ciratefi	Hits	140	140	140	140	140	700	114	171	120	118	89
	False pos.	0	0	0	0	0	0	1	18	0	3	2
	False neg.	0	0	0	0	0	0	2	16	0	2	31
SIFT	Hits	134	122	99	77	132	564	54	61	120	120	120
	False pos.	0	4	0	0	0	4	23	99	0	2	0
	False neg.	6	18	41	63	8	136	62	126	0	0	0
EasyMatch	Hits	131	*	*	*	*	*	101	118	118	66	93
	False pos.	9	*	*	*	*	*	15	69	2	54	27
	False neg.	9	*	*	*	*	*	15	69	2	54	27

Table 2
Simplification of Table 1, where we present the accuracy computed as Hits/(Hits+FP+FN).

	Toy shapes	McDonald	Buildings	Game A	Game B	Game C
Ciratefi	100%	97%	83%	100%	96%	73%
SIFT	80%	39%	21%	100%	98%	100%
EasyMatch	88%	77%	46%	97%	38%	63%

As each query image Q occurs only once in A, the results can be either correct (the algorithm correctly localizes the query image) or erroneous (the algorithm points an incorrect location or fails to locate the query image). Then, we repeated the same experiments using query images with 91×91 pixels. Both SIFT and Ciratefi searched for the templates knowing that each instance occurs only once in A. We specified in the Hough transform (that follows SIFT) to take the cluster with the maximal number of keypoint correspondences as the matching. With this choice, SIFT detects the template even if there is only one keypoint correspondence. The results are depicted in Table 3. Let us interpret the results:

Table 3

Error rates of Ciratefi and SIFT searching for 300 image patches in Mikolajczyk image database.

	41×41		91×91	
	Cirat.	SIFT	Cirat.	SIFT
Bark (zoom/rotation)	18%	69%	16%	16%
Bikes (focus blur)	23%	76%	10%	44%
Boat (zoom/rotation)	26%	54%	20%	27%
Graf (viewpoint)	50%	81%	44%	51%
Leuven (camera aperture)	8%	59%	4%	26%
Trees (focus blur)	38%	77%	54%	44%
UBC (compression)	25%	52%	24%	30%
Wall (viewpoint)	53%	46%	79%	30%
Average	30%	64%	31%	34%

Blur and JPEG compression: SIFT made far more errors than Ciratefi in set Bikes, because Ciratefi is more robust than SIFT against blurring. However, this superiority is not clear in set Trees, where both Ciratefi and SIFT made many errors. In this set, the 6 photos were taken in different instants and the tree leaves are waving in the wind, what in our opinion hinders *any* image matching algorithm from successfully finding small templates. According to the experiment with set UBC, Ciratefi is more robust than SIFT also against JPEG compression, because JPEG creates false edges weakening SIFT's performance.

Brightness/contrast: Ciratefi made far less errors than SIFT in set Leuven, because Ciratefi is fully brightness/contrast-invariant.

Perspective: Both Ciratefi and SIFT are not robust against perspective or affine deformation and so both made many errors in sets Graf and Wall. Set Wall has many small local textures that SIFT can use to find keypoint correspondences. So, SIFT was better than Ciratefi in this set, especially using large templates. On the other hand, set Graf has few textures and Ciratefi fared better.

Large templates: If a large template Q is given, *any* image searching algorithm can extract many small sub-templates T_1, ..., T_n from Q, search for them in image to analyze A, and can combine the results using Hough transform. This approach would result in a high hit rate for Q, even the hit rates are low for sub-templates T_1, ..., T_n. This is what SIFT followed by Hough transform is doing implicitly. So, SIFT fared relatively better using large templates.

3.3. Processing time

For A with 465×338 pixels, Q with 51×51 pixels, 6 scales and 36 angles, the complete Ciratefi algorithm took 12s, divided as follows: Cifi took 2s for the 3D matrix calculation and 4s for the correlation process; Rafi took 5s; Tefi took about 1s. EasyMatch and SIFT take respectively 1.5s and 2s to do the same task. These times were obtained in a 2GHz Intel Core 2 Duo. Moreover, SIFT is especially fast when searching for many different templates in an unchanging image A, because most of the processing time is spent in computing the keypoints and the features of A (that can be done only once). From the experimental data, we conclude that Ciratefi is, in many cases, more accurate than SIFT and EasyMatch but slower.

4. Improved Ciratefi

Experimental results show that the original Ciratefi is accurate. However, there are many parameters left to be adjusted by hand. In this section, we introduce some improvements to automatize the choice of the parameters.

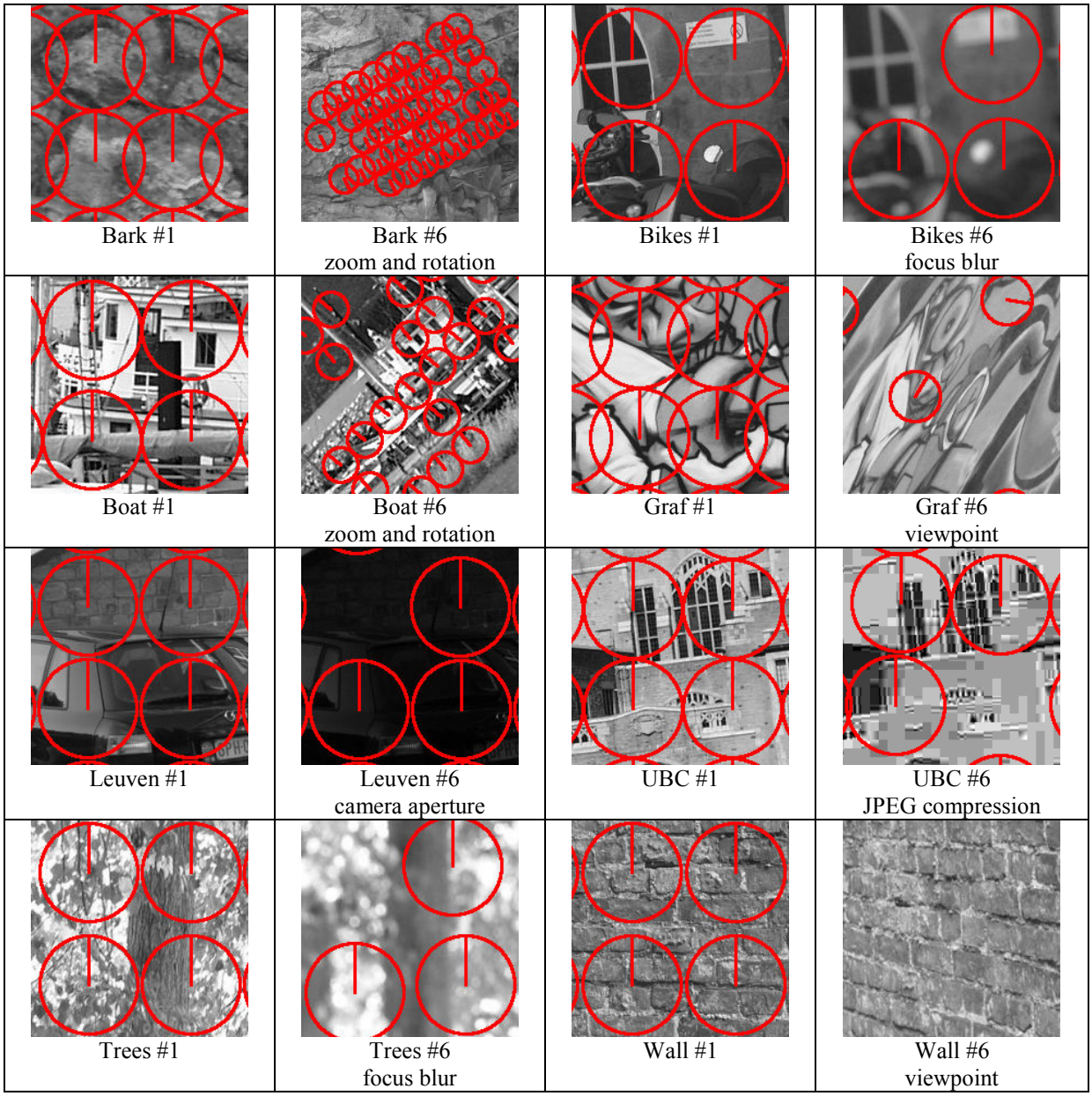

Fig. 5. We use Mikolajczyk database with 8 sets of natural images (each one with 6 progressively distorted images) to compare SIFT with Ciratefi. We took 50 square templates of the first image of each set and searched for them in the 6 images. SIFT uses the whole square template, while Ciratefi uses only the circular template inscribed in the square. The depicted images are parts of the Ciratefi output images, each one containing 50 red circles with pointers indicating the 50 matching positions found by Ciratefi.

4.1. Thresholds

The appropriate values for t_1, t_2 and t_3 are difficult to be set by hand. Choosing small thresholds (great number of candidate pixels) increases the accuracy but makes the process slower. Choosing large thresholds may discard true matchings. Moreover, the appropriate values of these thresholds use to vary according to the application. So, we replaced these parameters by p_1, p_2 and p_3, where p_i is the number/percentage of the candidate pixels. We also defined d_1, d_2 and d_3, where d_i is the distance that separates the candidate pixels of the degree i. Experimentally, we chose p_1=2%, p_2=1% and p_3=1 pixel, for applications where Q appears once in A. This means that 2% of pixels of A are first-grade candidate, 1% of pixels of A are second-grade candidate, and one single pixel is chosen as the final matching. If Q appears more than once, it is possible to define a minimal distance between matchings. For example, specifying p_3=4 pixels and d_3=50 pixels, the program will choose 4 matchings separated by at least 50 pixels.

4.2. Scale range

It is also difficult to choose the appropriate scale range. So, we decided to build a pyramidal structure for A. This structure is widely used to obtain scale-invariance, for example, in [24]. Figure 6 depicts an example of this structure, obtained by concatenating the original A with its reduced versions at scales 0.5, 0.25, etc. For the query image Q, we chose to use a set of 5 scales in geometric progression {0.5, 0.57, 0.66, 0.76, 0.87}. With this choice, Ciratefi becomes scale-invariant in the range [0.5, ∞]. For example, suppose that Q at scale 0.66 was detected in A at scale 0.25. This means that Q scale (1/0.25)×0.66=2.64 appears in original A. In our implementation, we took care to not detect the template in the boundaries of different scales of pyramidal A.

In Cifi, we chose to use 16 circles whose radii increase in arithmetic progression from zero to the radius of the query image at the greatest scale. With this choice, the query image at the smallest scale (0.5) has 9 circles inside, enough to compute the correlation with some precision. Query image Q must be large enough in order to have 16 distinct circles inside the query image at scale 0.87. This happens when the size of Q is larger than 39×39. In the experiments, we use query images with typical size 61×61. Too large query Q makes the algorithm slower without increasing the accuracy. In this case, we suggest extracting and finding a sub-image of Q or resizing down both Q and A. In Rafi, we chose to use 36 angles.

With these alterations and choices, we obtained an implementation where the standard parameters can be used in all experiments.

Fig. 6. The scale range expands to infinite by building a pyramidal structure for A. Left column is the original A and the right column is A at scales 0.5, 0.25, 0.125, etc. If a query Q matches a region in the right column (red circles), actually Q at larger scale matches a region in the original image A (green circles in the left column).

4.3. Structural similarity

Structural similarity (SSIM) index is an image distortion metric for grayscale images designed to emulate the human visual perceptual system [38]. It separates the image distortion in three independent components: luminance (or brightness), contrast and structure (or correlation coefficient). Then, each component receives a weight that depends on the application.

To assess the perceptual similarity between two images X and Y, the local statistics μ_x, μ_y, σ_x, σ_y, σ_{xy} are computed within the local windows $\boldsymbol{x}$ and $\boldsymbol{y}$ that moves pixel-by-pixel over the entire images (where μ_x is the mean of $\boldsymbol{x}$, σ_x is the standard deviation of $\boldsymbol{x}$ and σ_{xy} is the covariance of $\boldsymbol{x}$ and $\boldsymbol{y}$). The moving window can be an 8×8 square window or an 11×11 circular Gaussian weighted window. In each window, three similarity functions are computed:

- $$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \quad (12)$$
- $$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \quad (13)$$
- $$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \quad (14)$$

where l measures the lightness similarity, c measures the contrast similarity and s measures the structural similarity. The constants C_i are small numbers introduced to avoid numerical instability when the denominators are close to zero. The structural similarity is defined:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \left[l(\mathbf{x},\mathbf{y})\right]^{\alpha}\left[c(\mathbf{x},\mathbf{y})\right]^{\beta}\left[s(\mathbf{x},\mathbf{y})\right]^{\gamma} \quad (15)$$

where α, β and γ are parameters to adjust the relative importance of the three components. The overall similarity of the two images is defined as the mean SSIM.

A template matching uses some metric to measure how well template Q matches a region of image A around a pixel p. Mean square difference and correlation coefficient are two popular metrics used in template matching. As SSIM was designed to evaluate how perceptually similar are two image patches, it seems natural to use it as similarity measure in template matching. Using SSIM, the user can assign different weights for brightness, contrast and structure depending on the application. To obtain a complete invariance to brightness/contrast, one can set $\alpha=\beta=0$ and $\gamma=1$. However, this is not a good choice because regions of A with constant grayscale will match any template (SSIM will be one). In brightness/contrast invariant applications, we use $\alpha=\beta=0.01$ and $\gamma=0.98$. If brightness/contrast changes only slightly in an application, higher weights may be assigned to α and β to increase the accuracy.

In the improved grayscale Ciratefi, we replace the correlation coefficient by SSIM as the similarity measure in all the three filters.

As a curiosity, SSIM-based (not RS-invariant) template matching can become computationally efficient for finding rectangular templates using the same ideas of the fast normalized cross correlation [22].

5. Color-Ciratefi[6]

In this section, we introduce color-Ciratefi, which takes into account the color information. According to the literature, it seems that there is no color invariant good for all applications. Our experiments confirm this information, indicating that color is not always useful. Color can even degrade Ciratefi's accuracy in applications with great illumination variation. However, in most applications, color increases the accuracy. This increasing is more considerable for detecting templates with distinctive colors in an illumination-controlled environment.

5.1. The proposed similarity measure

Ciratefi's robustness is rooted in using correlation of local mean grayscales instead of gradient orientations. To be consistent with this concept, we do not use color invariants that takes into account only color gradient, like [14]. Instead, we use chromaticity itself that allows us to distinguish the color of an image patch (that is, the property that allows us to say that an object is yellow, red, green, etc.)

CIE L*a*b* (CIELAB) color space was designed to be perceptually uniform, that is, a small perturbation to a color value produces a change of about the same perceptual importance across the range of all colors. Moreover, CIELAB isolates the lightness L* from the chromaticity a*b*. So, this color space is especially suited to evaluate the similarity of two image patches, evaluating independently the lightness similarity and chromaticity similarity.

We made some simple experiments using images taken under different illumination conditions. We concluded that the chromaticity a*b* remains relatively constant under small changes of illumination temperature, intensity and direction. However, we also realized that under a severe illumination variation, the chromaticity a*b* changes considerably. We tested also some other color spaces concluding that CIELAB's chromaticity is one of the most stable.

In CIELAB space, the lightness L* varies from 0 to 100. The range of chromaticity components a*b* depends on the original color space of the image. If the original color space is RGB, one can assume the range -100 to +100.

Let $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ be two vectors of colors. Each component x_i or y_i is composed by a set of tristimulus values L*, a* and b* denoted, respectively, as x_{iL}, x_{ia}, x_{ib} and y_{iL}, y_{ia}, y_{ib}. The similarity functions l, c, and s are computed on the component L* as in the grayscale case. For similarity of chromaticity h, we use the Euclidean distance of components a* and b*, because it is typically used as the distance measure in CIELAB color space [5, 17]:

$$h(\mathbf{x},\mathbf{y}) = 1 - \frac{\sum_{i=1}^{n}\sqrt{(x_{ia}-y_{ia})^2 + (x_{ib}-y_{ib})^2}}{200\cdot\sqrt{2}\cdot n} \quad (16)$$

where 200 is the greatest possible difference between the components a* and b*. To obtain the similarity

[6] A preliminary work was published in [3].

measure, the distance is subtracted from one. We define the color structural similarity as:

$$\mathrm{CSSIM}(\mathbf{x},\mathbf{y}) = \left[l(\mathbf{x},\mathbf{y})\right]^{\alpha}\left[c(\mathbf{x},\mathbf{y})\right]^{\beta}\left[s(\mathbf{x},\mathbf{y})\right]^{\gamma}\left[h(\mathbf{x},\mathbf{y})\right]^{\delta} \quad (17)$$

We use weighted geometric mean (instead of weighted arithmetic mean between SSIM and *h*) because either complete chromaticity dissimilarity or complete lightness dissimilarity represents a complete dissimilarity of the two patches. In the improved color Ciratefi, we use CSSIM as the similarity measure in all the three filters.

In the remainder of this paper, we use $\alpha=\beta=0.01$ and $\gamma=\delta=0.49$ in all color experiments, to obtain a template matching invariant to brightness/contrast and that takes into account both color and structure information. For grayscale experiments, we use the improved grayscale Ciratefi with $\alpha=\beta=0.01$ and $\gamma=0.98$.

6. Experimental Results for Improved Ciratefi

6.1. Preliminary experiment

We made a preliminary experiment using toy shape images, where color is manifestly an useful information and where there are very few local textures (what makes SIFT disadvantageous over the other algorithms). In this experiment, we searched for 16 query shapes (with approximately 43×43 pixels) in 3 images to analyze with different rotations and scales. Figure 6 depicts one of the 3 images with the results of the 16 searchings superimposed.

We tested the improved color-Ciratefi[7], the improved grayscale Ciratefi, C-color-SIFT[8] [7], the grayscale SIFT, color EasyMatch and grayscale EasyMatch. All algorithms knew that there was only one instance of *Q* inside each *A*. The Hough transform that follows C-color-SIFT and SIFT was programmed to detect the template even if there is only one keypoint correspondence between *Q* and *A*. In EasyMatch, we set the range of scales from 50% to 200%. Table 4 presents the results. As expected, color-Ciratefi and color easyMatch had the lowest error rate (0 errors or 0%), followed by grayscale Ciratefi with 8 errors or 17%. Surprisingly, C-color-SIFT made twice more errors (26 errors or 54%) than grayscale SIFT (13 errors or 27%).

[7] http://www.lps.usp.br/~hae/software/cirateg

[8] http://staff.science.uva.nl/~mark/downloads.html#colorsift

Table 4

Errors rates of each algorithm searching for 48 toy shapes.

color Ciratefi	0%
gray Ciratefi	17%
C-color-SIFT	54%
gray SIFT	27%
color EasyMatch	0%
gray EasyMatch	40%

6.2. Mikolajczyk's image database

Next, we compared the algorithms using Mikolajczyk's image database. We tested EasyMatch 1.1 only in some color cases, due to the impossibility to run automated tests. We discarded the set Boat, because it is originally grayscale. In all other sets except Bark, we reduced the images to 50% of the original sizes, extracted twenty 61×61 templates uniformly distributed in the first image and searched for them in the six reduced images. In set Bark, we reduced the first image to 50% of the original size, extracted twenty 61×61 templates uniformly distributed in the first image and searched for them in the six original non-reduced images. The results are in Table 5.

Overall, the two Ciratefis made fewer errors than the two SIFTs. Surprisingly, grayscale Ciratefi and color-Ciratefi made the same number of errors. C-color-SIFT made considerably more errors than grayscale SIFT. EasyMatch has the highest error rates. Some considerations:

Great illumination change: In set Leuven, there is great brightness/contrast variation and so the color algorithms made considerably more errors than the respective grayscale versions.

Colorful images: Sets Graf, Trees and UBC have some colorful patches and almost no illumination change. In these 3 sets, color-Ciratefi made fewer errors than the grayscale version. Surprisingly, even in these cases, C-color-SIFT made more errors than the grayscale version.

Blur and JPEG compression: The two Ciratefis made far less errors than the two SIFTs in sets Bikes and Trees (focus blur). In blurred images, SIFTs extracts only a small amount of keypoints, yielding errors. Ciratefis also made far less errors than SIFTs in set UBC (JPEG compression). In this case, a large amount of inconsistent features arises from the artifacts introduced by the JPEG compression, leading to erroneous SIFT keypoint matchings.

Table 5

Error rates of each algorithm searching for 120 patches of Mikolajczyk database. * means that the experiment was not done.

	color Cirat.	gray Cirat.	color SIFT	gray SIFT	color EMat.
Bark (zoom/rotation)	0%	0%	1%	0%	*
Bikes (focus blur)	0%	0%	28%	31%	53%
Graf (viewpoint)	33%	38%	56%	55%	*
Leuven (cam. apert.)	13%	3%	47%	23%	55%
Trees (focus blur)	10%	13%	38%	38%	86%
UBC (JPEG)	2%	3%	54%	13%	49%
Wall (viewpoint)	30%	30%	27%	33%	*
Average	12%	12%	36%	27%	60%

6.3. ALOI image database

ALOI is a color image collection of small objects[9] [15]. In order to capture the sensory variation in object recordings, the authors systematically varied viewing angle, illumination angle, and illumination color for each object.

We took the images with 4 different illumination colors (with the illuminating lamp temperatures 3075K, 2750K, 2475K and 2175K) of the first 20 objects, reduced them by 2 and glued the images with the same illumination temperatures together, obtaining four 288×480 images. We searched the objects in the first image (3075K), cropped to 61×61 pixels, in the four images, using the four algorithms. The errors are depicted in row Color-A of Table 6. We repeated the experiment using the next 20 objects (row Color-B).

We took the images with 4 different illumination directions (identified as l8c1, l7c1, l6c1, l4c1 in the database) and searched the objects in the first image (l8c1) in the four images, obtaining rows Illum-A and Illum-B.

We searched the unrotated objects in images with the objects rotated in 4 different angles (0, 20, 40 and 60 degrees), obtaining rows View-A and View-B. Figure 7 depicts two of the images obtained in this experiment.

We searched the unblurred objects in images distorted with Gaussian blur with kernels 1×1, 3×3, 5×5 and 7×7 (σ=0, 0.95, 1.25 and 1.55), obtaining rows Blur-A and Blur-B.

[9] http://staff.science.uva.nl/~aloi/

We searched the uncompressed objects in JPEG-compressed images with qualities 100%, 75%, 50% and 25%, obtaining rows JPEG-A and JPEG-B.

Table 6

Errors rates obtained searching for 80 objects in ALOI database. * means that the experiment was not done.

	color Ciratefi	gray Ciratefi	color SIFT	gray SIFT	color EMatch
Color-A	0%	1%	10%	5%	60%
Color-B	0%	0%	15%	8%	*
Illum-A	6%	13%	30%	38%	53%
Illum-B	15%	26%	35%	48%	*
View-A	21%	29%	39%	53%	65%
View-B	20%	24%	36%	56%	*
Blur-A	0%	0%	28%	31%	*
Blur-B	0%	0%	26%	28%	*
Jpeg-A	0%	3%	51%	19%	*
Jpeg-B	0%	0%	46%	14%	*
Average	6%	9%	32%	30%	59%

Let us analyze the results of Table 6. Color-Ciratefi has equal or lower error rate than grayscale Ciratefi in all tests, indicating that the use of color helps to find colorful objects. The grayscale Ciratefi has lower error rates than the best SIFT in all cases, and this superiority is especially evident in: illumination color variation, blurring and JPEG compression. Color EasyMatch was the worst algorithm in all tests.

To find an object, typically color-Ciratefi takes 13s; grayscale Ciratefi takes 9s; C-color-SIFT takes 3s; grayscale SIFT takes 2s and EasyMatch less than 1s.

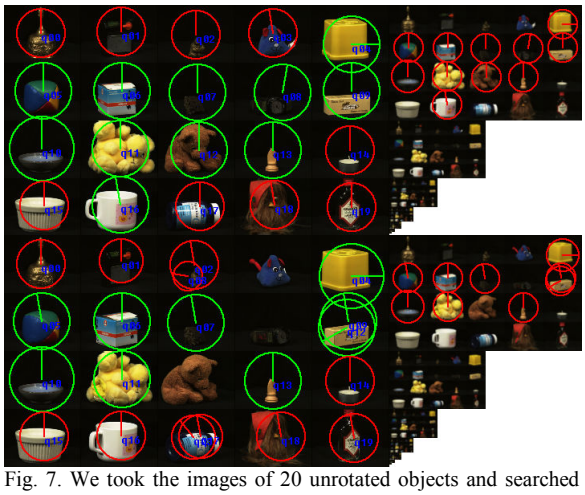

Fig. 7. We took the images of 20 unrotated objects and searched for them in images of the objects rotated by 20°, applying Ciratefi 20 times. The image at top depicts the result, with 1 error. We repeated the process to objects rotated by 40°, resulting in the image at bottom, with 4 errors. The red circles indicates the matchings of Q at scales {0.5, 0.57, 0.66, 0.76, 0.87}. Whenever Q matches a reduced A (in red in the right column), actually Q at a larger scale matches the original A (in green in the left column).

7. Considerations

Our experiments indicate that Ciratefi is more accurate than SIFT and EasyMatch most of the time. Nevertheless, SIFT has many practical advantages over Ciratefi:

- SIFT is faster than Ciratefi. SIFT is especially fast when searching for many different templates in an unchanging image A, because most of the processing time is spent in computing the keypoints and the features of A (that can be done only once).
- The query image in Ciratefi must contain only the searching pattern. The query image in SIFT may contain the searching pattern among many other "junk" background patterns, because it searches the occurrences of the keypoints of Q in A instead of the whole query image.
- Template matching using SIFT followed by Hough transform is robust to partial occlusions, while Ciratefi by itself is not.
- SIFT is wholly scale-invariant, while even the improved Ciratefi is scale-invariant only from 0.5 to ∞. As a truly scale-invariant method, SIFT can find small or large template. Meanwhile, Ciratefi is better suited for finding relatively small templates, because large template may be time-consuming.

In our opinion, even if Ciratefi is not practical right now for some applications, it deserves to be more thoroughly studied because of its superior accuracy. There remains the challenge of designing an algorithm as practical as SIFT and as accurate as Ciratefi.

We have already taken some steps in that direction. Ciratefi repeats exactly the same series of simple operations for each pixel, making it especially appropriate for highly parallel implementation. One of the authors has participated of a research [29] to implement Ciratefi in FPGA (Field Programmable Gate Array). In this research, the authors have simulated the first of the three Ciratefi filters and concluded that the hardware implementation is 5000 times faster than the software implementation and can classify one pixel as candidate or non-candidate in each clock (after the initial latency).

One of the authors also have presented a template matching based on circular and radial projections that makes use of FFT (Fast Fourier Transform) and is fast even in a conventional computer [18].

8. Conclusions

In this paper we have presented an RST-invariant template matching named Ciratefi, with controlled robustness to brightness/contrast changes. We have compared Ciratefi with SIFT and EasyMatch concluding that our technique is, most of the time, more accurate but slower. As the original Ciratefi has many adjustable parameters, we have presented a methodology to automatize the choice of all parameters. We have also introduced a version of Ciratefi for color images. We have compared color-Ciratefi, grayscale Ciratefi, C-color-SIFT and grayscale SIFT. The overall result indicates that color-Ciratefi is the most accurate algorithm in most situations. However, in some applications with large illumination variation, grayscale Ciratefi can overperform the color version. In many applications, SIFT is more useful in practice than Ciratefi due to its many nice properties. However, as we have demonstrated experimentally that Ciratefi is manifestly more accurate than SIFT under many common image distortion scenarios, there remains the challenge for designing an algorithm as practical as SIFT but as accurate as Ciratefi.

References

[1] A. E. Abdel-Hakim and A. A. Farag, CSIFT: A SIFT descriptor with color invariant characteristics, In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 1978-1983.

[2] C. Ancuti and P. Bekaert, SIFT-CCH: Increasing the SIFT distinctness by color co-occurrence histograms, In: Proceedings of 5th International Symposium on Image and Signal Processing and Analysis, 2007, pp. 130-135.

[3] S. A. Araújo and H. Y. Kim, Color-Ciratefi: A color-based RST-invariant template matching algorithm, In: Proceedings of 17th Int. Conf. Systems, Signals and Image Processing, 2010, pp. 101-104.

[4] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognition, vol. 13, n. 2, 1981, pp. 111-122.

[5] B. Bascle, O. Bernier and V. Lemaire, Illumination-invariant color image correction, In: Proceedings of Int. Workshop on Intelligent Computing in Pattern Analysis/Synthesis (IWICPAS), LNCS, vol. 4153, 2006, pp. 359-368.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, SURF: Speeded Up Robust Features, Computer Vision and Image Understanding, vol. 110, n. 3, 2008, pp. 346-359.

[7] G. J. Burghouts and J. M. Geusebroek, Performance evaluation of local colour invariants, Computer Vision and Image Understanding, vol. 113, n.1, 2009, pp. 48-62.

[8] Q. Chen, M. Defrise, and F. Deconinck. "Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(12):1156–1168, December 1994.

[9] M.S. Choi, W.Y. Kim, A novel two stage template matching method for rotation and illumination invariance, Pattern Recognition 35(1) (2002) 119-129.

[10] H. Chong, S. Gortler and T. Zickler, A Perception-based Color Space for illumination-invariant image processing, ACM Trans. on Graphics, vol. 27, n. 3, 2008, pp. 1-7.

[11] B. Cyganek, Circular road signs recognition with soft classifiers, Integrated Computer-Aided Engineering 14 (2007) 323-343.

[12] B. V. Funt and G. D. Finlayson, Color Constant Color Indexing, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 17, n. 5, 1995, pp. 522-529.

[13] B. V. Funt, K. Barnard and L. Martin, Is machine colour constancy good enough?, In: Proceedings of the 5th European Conference on Computer Vision, LNCS, Vol. 1406, 1998, pp. 445-459.

[14] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders and H. Geerts, Color invariance, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 23, n. 12, 2001, pp.1338-1350.

[15] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, The Amsterdam library of object images, Int. Journal of Computer Vision, vol. 61, n. 1, 2005, pp. 103-112.

[16] T. Gevers and A. Smeulders, Color based object recognition, Pattern Recognition, vol. 32, 1997, pp. 453-464.

[17] A. Gijsenij, T. Gevers and M. P. Lucassen, A perceptual comparison of distance measures for color constancy algorithms, In: Proceedings of the 10th European Conference on Computer Vision, LNCS, vol. 5302, 2008, pp. 208-221.

[18] H. Y. Kim and S. A. de Araújo, Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast, In: Proceedings of the 2nd Pacific Rim conference on Advances in image and video technology, LNCS, vol. 4872, 2007, pp. 100-113.

[19] H. Y. Kim, Rotation-discriminating template matching based on Fourier coefficients of radial projections with robustness to scaling and partial occlusion, Pattern Recognition, vol. 43, n. 3, 2010, pp. 859-872.

[20] Y. Lamdan and H. J. Wolfson, Geometric hashing: a general and efficient model-based recognition scheme. In: Proceedings of 2nd Int. Conference on Computer Vision, 1988, pp. 238-249.

[21] T. K., Leung, M. C. Burl, and P. Perona, Finding faces in cluttered scenes using random labeled graph matching. In: Proceedings of 5th Int. Conference on Computer Vision, 1995, pp. 637-644.

[22] J. P. Lewis, Fast normalized cross-correlation, Vision Interface, 1995, pp. 120-123.

[23] Y. Lin and C. Chen, "Template matching using the parametric template vector with translation, rotation and scale invariance," *Pattern Recognition*, vol. 41, no. 7, Jul. 2008, pp. 2413-2421.

[24] D. G. Lowe, Distinctive image features from scale-invariant keypoints, Int. Journal of Computer Vision vol. 60, n. 2, 2004, pp. 91-110.

[25] D. Marimon and T. Ebrahimi, Efficient rotation-discriminative template matching, Lecure Notes in Computer Science, vol. 4756, 2008, p. 221-230.

[26] K. Mikolajczyk, A performance evaluation of local descriptors, IEEE Trans. on Pattern Analysis and Machine Intelligence , vol. 27, n. 10, 2005, pp. 1615-1630.

[27] T. Moerland and F. Jurie, Learned color constancy from local correspondences, In: Proceedings of the 2005 IEEE Int. Conference on Multimedia and Expo, 2005, pp. 820-823.

[28] F. Mokhtarian, A. K. Mackworth, "A theory of multi-scale, curvature based shape representation for planar curves," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 14, n. 8, 1992, pp. 789-805.

[29] H. P. A. Nobre and H. Y. Kim. Automatic VHDL generation for solving rotation and scale-invariant template matching in FPGA. In: Proceedings of 5th Southern Programmable Logic Conference, 2009, pp. 229-231.

[30] K. A. Raftopoulos, N. Papadakis, K. Ntalianis and S. Kollias, A visual pathway for shape-based invariant classification of gray scale images, Integrated Computer-Aided Engineering 14 (2007) 365–378.

[31] S. Reddy and B. N. Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration". IEEE Trans. on Image Processing, 3(8):1266–1270, August 1996.

[32] A. Sajjanhar, G. Lu, D. Zhang, J. Hou, W. Zhou, and Y.P. Chen (2010), "Spectral Shape Descriptor Using Spherical Harmonics," *Integrated Computer-Aided Engineering*, 17:2, pp. 167-173.

[33] K. E. A. van de Sande, T. Gevers and C. G. M. Snoek, Evaluation of color descriptors for object and scene recognition. In: Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008 , pp. 1-8.

[34] G. Schaefer, How useful are colour invariants for image retrieval, In: Proceedings of 2nd Int. Conference on Computer Vision and Graphics, 2004.

[35] M. J. Swain and D. H. Ballard, Color Indexing, Int. Journal of Computer Vision, vol. 7, n.1, 1991, pp. 11-32.

[36] D. M. Tsai and Y. H. Tsai, Rotation-invariant pattern matching with color ring-projection, Pattern Recognition, vol. 35, n. 1, 2002, pp. 131-141.

[37] F. Ullah, and S. Kaneko: Using orientation codes for rotation-invariant template matching. Pattern Recognition, vol 37, n. 2, 2004, pp. 201-209.
[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. on Image Processing, vol. 13, n. 4, 2004, pp. 600-612.